# String formatting

I decided that I wanted the player to be able to input commands to the console as if they were talking to a person as I believed this made games such as Zork (1979) and Deeper (2016) easier to learn to play and therefore more immersive. To be able to use the input I would have to split it into smaller string representing verbs, items and exits.

Firstly, I split the inputted string into a list of separate words and removed all of the conjunctions, prepositions and other unnecessary words from that list to leave me with only the verbs and nouns (such as item names and exits). A *for loop* running through a list of “*removable*” words was used to remove all of the conjunctions and prepositions that I had defined in the class. Later in the development of the algorithm, I wanted the items in the game to be able to have names containing spaces like in the games I had researched. This added another step to the string formatting method which involved stitching individual strings together between certain “*splitter*” words such as “with” and “and”. Once the strings are all stitched back correctly the *splitter* words are all removed the same way as the *removable* words are

# Command Parsing

After formatting, I am left with a list of strings that can contain verbs, item names and exits, I can now start to figure out what the player wants to do and from this, attempt to call the right method for them.

The first check I make is to see whether there are more than one words in the command list. If there is only one word, I firstly check to see if it’s an exit in the current location, if so then the moveToLocation method is called and I return out of the method. However, if the command wasn’t a direction then it is put into a switch statement checking if it’s a look, show inventory or a help command, calling their own methods respectively.

Commands containing more than one word go through the *Complex command* part of the algorithm. Here I check to find a verb and set the correct command type for that verb, a dictionary is used here to pair string representations of the verbs to their own enumerated versions, this allow me to easily allow multiple similar verbs to call the same method such as “move” and “walk” both setting the *Move* command type. Once the command type has been found, it is used by another switch statement to pass the remaining strings in the *command list* as arguments in the matching methods.

The idea of searching for a verb in the input comes directly from Zork which, if it cannot find one will tell you directly it doesn’t understand the verb you entered.

# Look

# Walk/move

The move method is split up into two methods to allow an input of a string as well as the exit directally, the method taking in the string eventually calls the other one I’ll explain that one first. Firstly, I am looking for an exit in the current location matching the given string, this is done by using the List<>.Find method with a lambda expression which compares all of the *ToString* values of the elements to the given string. Before learning about lambda expressions I had been using *foreach* loops to go through the list and individually check the names against the string which became rather messy, the way I search for the exit now saves the method from getting too complicated and messy. Finally, once the exit is found we pass it into the other move method; we check that the exit is not locked, by seeing if the exit’s Key property is null. Now all I do is set the *current location* to the exit’s location and return out of the method; the game then updates and displays the new location.

# Use

The use command allows the player to unlock an exit with a Key item they have picked up, to allow the use of keys in the game I added a key variable to the Exit class where if an item is present, the exit is locked and alternatively, a null item means that the exit is unlocked.

Two things are required whn *using* items, the item

# Bibliography

The Pixie. (2016). *Deeper*. [Video Game - Online]. Available at: <http://textadventures.co.uk/games/view/em15b32xd0o-y-ysvgrtcg/deeper> [Accessed 24 Feb 2016]

Anderson, T., Blank, M., Leibling, D. and Daniels, B. (1979). *Zork*. [Video Game] Infocom. Available at: <http://textadventures.co.uk/games/view/5zyoqrsugeopel3ffhz_vq/zork> [Accessed 24 Feb 2016]